CoReDac: Collision-Free Command-Response Data Collection

Thiemo Voigt, Fredrik Österlind
Swedish Institute of Computer Science
thiemo@sics.se,fros@sics.se

Abstract

Most of the existing sensor network deployments are convergecast data collection applications that transmit data from multiple sources to a sink. In this paper, we present CoReDac, a self-organizing, collision-free convergecast protocol. In contrast to previous solutions, CoReDac consists not only of a data collection phase but also an efficient and collision-free command phase. This way, the protocol can be used below the application layer for wireless versions of building automation protocols such as BACnet that require both efficient response and command phases. We present experiments that show that CoReDac works as expected and demonstrate its energy-efficiency for low duty cycle command-response applications by comparing it to X-MAC.

1 Introduction

Building Automation Systems (BAS) are used to both improve the indoor climate in buildings and to reduce the operational costs. Originally, most BAS were heating, ventilation and air-conditioning (HVAC) systems. To further increase management and reduce costs, other functions such as lighting, safety, security, and transportation supervision have been integrated into BAS. Traditionally, BAS have been used in large buildings, for example, schools, hospitals, and offices. For these types of buildings, the construction costs constitute only one seventh of the overall operational costs [12]. Therefore, reducing the operational costs with a BAS provides a great potential for savings.

Wireless sensor networks (WSN) consist of networks of tiny embedded systems with sensing capabilities that communicate with each other using an on-board low power radio module. Typically, sensor nodes transport measured data to a base station in a multi-hop fashion. Most available sensor nodes are matchbox-sized and powered by batteries. Since it is in general not possible, or too labor-intensive, to replace the nodes' batteries, reducing power consumption is one of the major research activities in the area of wireless senor networks. Since typically wireless communication is the most energy-intensive task sensor nodes perform [21], a particular focus has been on power-efficient communication protocols for wireless sensor networks.

Integrating WSN and BAS has a number of advantages. The main advantage is that WSN avoid wiring and hence reduce the installation costs compared to wired solutions. It is further possible to extend an existing BAS with wireless sensors in order to increase the sensor coverage. As the installation costs decrease, it is possible to increase the number of sensors and hence the spatial resolution. The increased spatial resolution allows for more fine-grained measurements and control. A further advantage is that wireless technology enables temporary measurements: a network can be set up to perform measurements during a limited time in order to measure, optimize and evaluate the effect of the optimization. Moreover, wireless sensors can also be installed more easily in unapproachable places such as at high heights.

There are efforts to use ZigBee as the underlying layers for BAS systems [19]. ZigBee requires that routers, i.e. the nodes that forward packets on behalf of other nodes, are mains-powered. In order to leverage the real advantage of using WSNs, routers should not be mains-powered. The radio is a sensor node's major energy consumer and idle listening requires almost as much power as receiving or transmitting data. Hence, efficient solutions must allow routers to turn off their radio as often as possible which in multi-hop wireless networks requires protocols that enable sensor nodes to turn off their radio in a synchronized fashion. Many existing sensor network applications collect data from sensor nodes and transmit the data in a multi-hop fashion to a base station. This task can be performed by convergecast protocols. Building automation protocols such as BACnet [1] are centered around a command-response paradigm where a command is sent to a device and a reply is awaited. Therefore convergecast protocols alone are not sufficient in the context of building automation. However, the most prominent of convergecast protocols, D-MAC [13] and Dozer [3], have been optimized for data collection only and do not offer efficient solution for disseminating commands from the base station to the sensor nodes.

In this paper, we present CoReDac. CoReDac constructs a collision-free tree that allows for low-power command-response data collection. One of our basic ideas is to use acknowledgements for slot assignment and syn-

chronized node wake-up scheduling. WiseMAC has exploited a similar idea by including the sampling schedule offset into acknowledgements [8]. CoReDac's ability to synchronize node wake-up without explicit time synchronization seems very appealing and useful for data collection applications that demand a low duty cycle and long lifetime. CoReDac's capability of performing efficient command-response convergecast makes it suitable as an underlying layer for e.g. BACnet and substantially extends the lifetime of battery-powered sensing devices in applications such as building automation. We present CoReDac and experiments on real hardware that validate the design of our protocol and demonstrate its energy-efficiency.

We have presented some of the ideas described here in a poster paper [26]. CoReDac also includes multi-channel support that is described elsewhere [27]. The new contributions of this paper include the detailed description of CoReDac's slot assignment procedure as well as the command phase that we introduce and evaluate in this paper. To the best of our knowledge CoReDac is the first protocol for collision-free command-response convergecast.

The rest of our paper is outlined as follows: In the next section we present CoReDac's design. Section 3 briefly presents our implementation. Section 4 is devoted to experimental results on real hardware. Before concluding we discuss related work in Section 5.

2 CoReDac Design

In this section, we present the design of CoReDac. We start by presenting the design of the data collection phase, i.e. the phase during that the data flows from the sources to the base station. In the second part, we discuss the command phase, i.e. the phase during that commands or code updates are disseminated to the sources, a process that is triggered by the base station.

2.1 Design of the Collection Part

CoReDac uses the notion staggered slots introduced by DMAC [13].

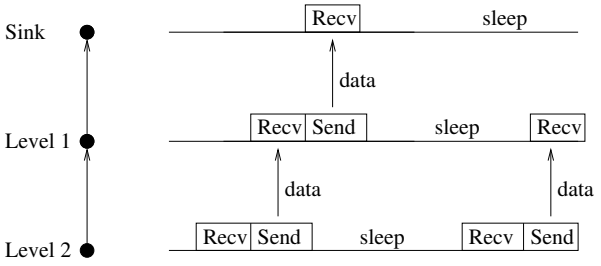

Figure 1. Staggered wake-up a la DMAC

Figure 1 presents the staggered wake-up scheme employed by DMAC. As shown in the figure, different levels of the tree send at different times in order to reduce delay and contention. However, in DMAC there is still contention between nodes on the same level.

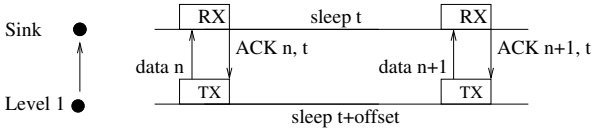

Figure 2. Basic synchronization scheme

Figure 2 presents one of CoReDac's basic ideas. The receiver of a message sends an ACK that besides acknowledging a packet also states in how many seconds it will turn on its radio again and is ready to receive packets from its children. We call this time $sleep\_time$ and assume that it is determined by the sink node. This basic scheme does not require explicit time synchronization since only relative time is of importance. Note that the figures are simplified in that nodes do not need to have their radio turned on during the whole duration of their TX slots.

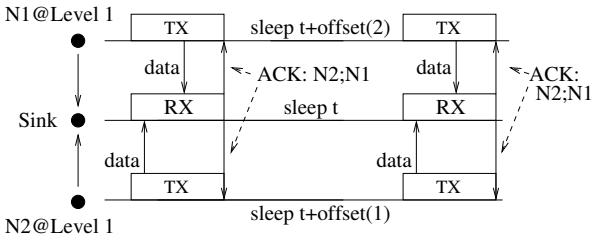

Figure 3. On-demand slot assignment that avoids collisions

By adding offsets for different nodes into the ACK packet, we can extend the basic scheme to let a parent node assign slots to its children. Figure 3 demonstrates how the sink assigns different offsets to its children. Each acknowledgement packet contains ACK-fields denoting the positive and negative acknowledgement of the children's data packets in always the same order. This way, the position of the ACK-field can be used by the children to compute their offset into the parent's RX slot. In the scenario in Figure 3, node $N2$ may send before $N1$. The parent's RX slot must be long enough to allow a maximum number of children to transmit. Furthermore, we must allow new nodes to join the tree. In order to reduce problems due to collisions between several joining nodes, the latter randomly choose one of the extra slots that we provide.

The scheme can be applied recursively to extend it towards a whole tree. However, when extending the scheme to several levels we need to take care to avoid collisions between nodes on different levels. Towards this end, we introduce a maximum number of children per node. Based on this maximum and its position in the tree, a node can compute its listen and send slots as well as offsets for its children. This way, we build a collision-free tree with-

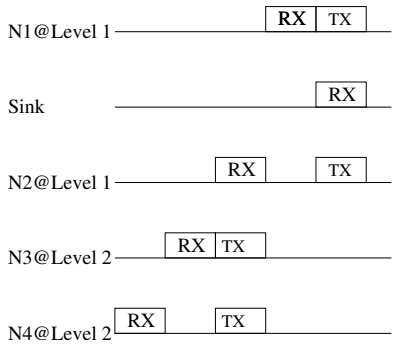

Figure 4. Slot assignment for larger tree

out explicit time synchronization. An example is shown in Figure 4. In this figure, $N2$ is the parent of $N3$ and $N4$. From the information in the acknowledgement, $N3$ and $N4$ can infer their RX slot and their offset into $N2$'s RX slot. The example of $N4$ shows that a node's RX and TX slot are not necessarily aligned, i.e. $N4$'s RX slot is not immediately followed by its TX slot. Hence, a node needs to turn its radio on and off more often which implies a little energy overhead that should be well compensated for by avoiding collisions.

Note that CoReDac assumes that the number of packets traveling from the sources to the sink does not increase when the packets come closer to the sink implying that data can be aggregated or not all nodes have data to transmit.

Computing listen slots The listen slot of a node depends on its position in the tree. If a node is at position pos, its RX slot is $pos - 1$ time slots before the base station's RX slot. For example, node $N4$'s RX slot in Figure 4 is two slots before the RX slot of its parent $N2$. Each node can compute its position based only on the knowledge of its parent's position, the parent's level and its position among its siblings, i.e. its offset into the parent's RX slot. All this information is available in the acknowledgement.

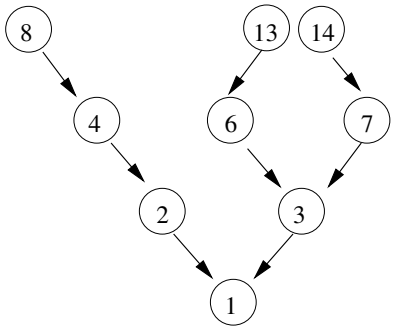

Figure 5. Computing listen slots

To compute the number of RX slots between a node and its parent, we add the number of nodes in the parent's level that have a position numbered higher than the parent (number of nodes on the same level and to the right of the parent in Figure 5) to the number of nodes in the node's level with a position numbered lower than the node (number of nodes to the left).

$$pos = parent\_pos + max\_pos(parent\_level) - parent\_pos + MAX * (parent\_pos - min\_pos(parent\_level)) + my\_pos \quad (1)$$

where my_pos is the node's position among its siblings that can be inferred from information in the acknowledgement.

$$max\_pos(parent\_level) = \sum_{0}^{parent\_level} MAX^{parent\_level} \quad (2)$$

$$min\_pos(parent\_level) = max\_pos(parent\_level - 1) + 1 \quad (3)$$

For example, consider how node 13 in Figure 5 computes its position based only on the knowledge of its parent's position, the parent's level and its offset with the help of the formulas above: $pos(13) = 6+7-6+(6-4)*2+2 = 13$.

2.2 Design of the Command Part

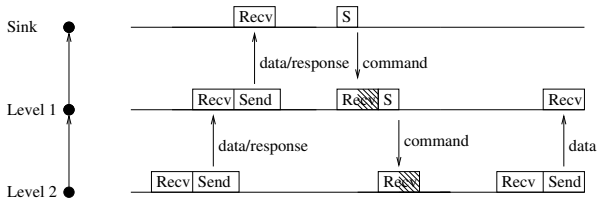

Figure 6. Design of command part

The command part of CoReDac is initiated by the base station when it sends a command to its children. The children then forward the command to their children as shown in Figure 6. It is possible to place this command part directly after the collection phase which gives the system the possibility to immediately react to the collected data. One could imagine that a system might activate a sprinkler if a fire is detected. It is also possible to use the command phase just before the collection phase which would allow the system to disseminate commands into the network and receive an answer fast. This corresponds very well with the client-server communication-model used in BACnet [18].

In both cases it is important, that the phases do not overlap and cause collisions. Avoiding overlap is no problem if the maximum depth of the tree is known. Note that it is also possible to have two command phases, one just before and one just after the collection phase.

RX slots for the command phase In the following we describe how we derive the RX time slots for the command phase.

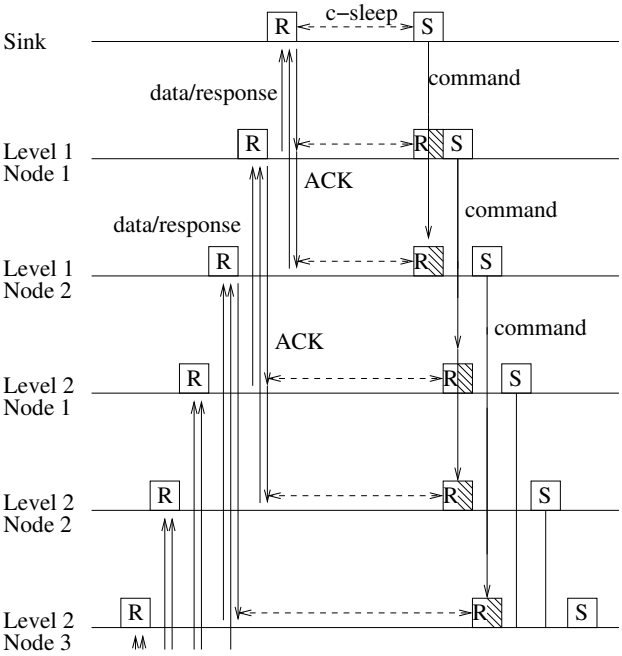

Figure 7. Computing slots for command phase

Figure 7 shows one collection and one command phase. As the figure shows, the RX time slot for the command phase starts at the same time for all children of a certain parent.

$$(ppos - 1 + ppos - 1) * RX - time + c\_sleep \quad (4)$$

Equation 4 denotes how a node computes the start time for its RX slot relative to the time it sent the last acknowledgement as shown in Figure 7. In this equation c_sleep denotes the sink's offset time between receiving data and sending a command. We must avoid overlap between the collection and the command phase since this might lead to collisions. In order to avoid overlapping the following must be true: $sleep\_time > c\_sleep + 2 * (RX - time * \#nodes) = c\_sleep + 2 * RXslottime * (\sum_0^{max\_level} MAX^{level} - 1)$.

The start time of an RX slot in the command phase is relative to the reception of the ACK in the collection phase. In the command phase, each node receives only exactly one packet, namely the one from its parent. Hence, a node can stop listening for commands when it has received the command message from its parent. In Figure 7 these are the shaded parts of the RX slots in the command phase.

$$(my\_pos - ppos - 1) * RX - time + RX - time/2 \quad (5)$$

Equation 5 denotes how a node computes the time for forwarding its command message relative to the end of the RX slot for reception of the command message.

3 Implementation

We have implemented CoReDac in the Contiki operating system [5] above the broadcast layer of the Rime protocol stack [6], i.e. we turn the radio on and off at the application layer. The scheme could also be implemented in the MAC layer below the Rime stack. Our current implementation is not optimized in that it does not try to minimize the guard times of the RX slots. In general, longer RX slots allow for larger clock drift. Meier et al. discuss how to minimize slot times [15].

4 Evaluation

In this section we present results from experiments with real hardware using the Tmote Sky platform [20] as well as simulations with the COOJA simulator [17].

4.1 Energy-efficiency of the Collection Phase

We measure the energy consumption of CoReDac's collection phase using four nodes deployed in a chain. In CoReDac, we achieve this by simply setting the maximum number of children per node to one. The energy consumption is measured using Contiki's software-based on-line energy estimation method [7].

We concentrate on the energy for radio listening as radio listening is the dominating factor for power consumption in WSNs [7]. We compare our protocol to X-MAC [2], a power-saving MAC protocol that is designed to run on top of the IEEE 802.15.4 physical layer. X-MAC reduces the power consumption by switching the radio on and off at regular intervals. When sending packets, nodes broadcast a train of short strobe packets. The strobe packet train is long enough to allow potential receivers to receive at least one strobe. For unicast packets, the strobe packets include the address of the receiver of the full packet. Having received a unicast strobe, a receiver directly sends a short acknowledgment packet. The sender can then immediately transmit the full packet. Other nodes that overhear the strobe packets can turn off their radio until the full packet has been transmitted.

Figure 8 shows the average power consumption for radio listening comparing X-MAC with CoReDac. Since our protocol has a constant radio listening time for each packet, the radio listening time decreases approximately linearly when less packets are sent. In all scenarios, CoReDac performs better than X-MAC. We have also measured the power consumption for transmitting packets. For CoReDac, this is less than around 1% of the power consumption for listening and receiving packets which confirms the measurements by Dunkels et al. [7]. The power consumption for transmitting with X-MAC is

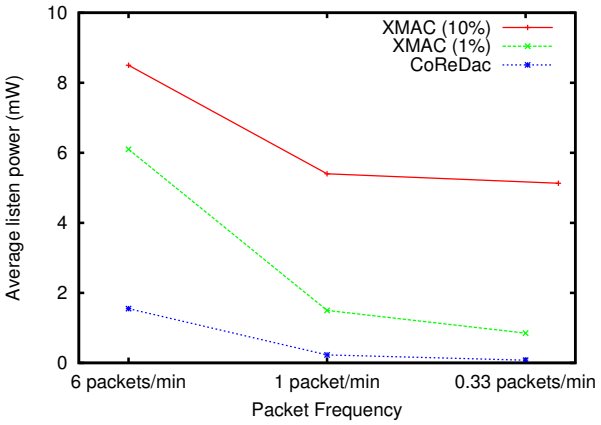

Figure 8. CoReDac's average radio listen power is lower than X-MAC's

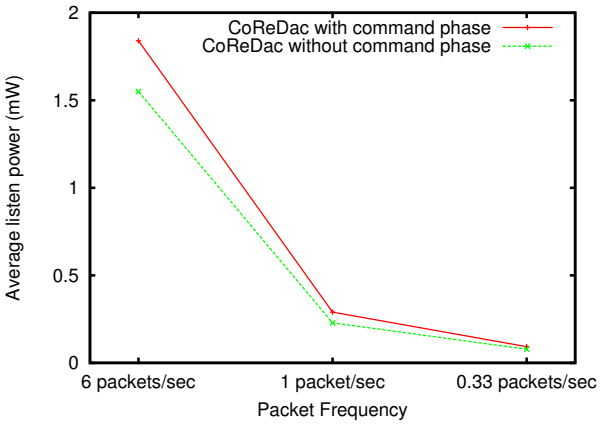

Figure 9. CoReDac's command phase is energy-efficient

slightly more expensive due to the transmission of the strobe packets.

We have confirmed the results for larger networks of up to 30 nodes using the COOJA sensor network simulator [17]. COOJA can simulate sensor networks at different levels and using the built-in sensor node emulator MSPSim [9] we have verified that the difference in power consumption between simulation and the results on real hardware in Figure 8 is less than 2%. This suggests that the results for larger networks obtained by simulation are realistic.

Figure 8 also shows that with the same duty cycle, X-MAC consumes less energy when there is less traffic, i.e. a duty cycle of 10% will not per se extend the lifetime of the network with a factor of 10 but that the lifetime extension depends on the traffic volume. The reason for this is that after the intended receiver of a packet has indicated that it is ready to receive a packet, the receiver must have its radio turned on until it has received the packet. This task consumes much more energy than listening for the short X-MAC strobe packets.

The results indicate that CoReDac is suitable for data collection applications with very low duty cycles, for example applications collecting temperature values in buildings.

4.2 Energy-efficiency of CoReDac with Command Phase

In the experiment in this section, we evaluate the energy efficiency of CoReDac's command phase using the same setup as in the previous section, i.e. we deploy four nodes in a chain.

The results are shown in Figure 9. The figure shows that the additional energy consumption required for the command phase is quite low, namely around 20%. The energy consumption for the command phase is lower than for the response phase since in the command phase, a node expects exactly one packet transmitted by its parent and can turn off the radio as soon as it has received this packet. Furthermore, nodes do not need to listen to new nodes that want to join the network.

4.3 Impact of the maximum number of children

In order to construct collision-free trees, we need to define a maximum number of children per node. This maximum number of children impacts the minimal $sleep\_time$, i.e. the period between two collection phases an operator can set. The maximum length of the period depends on the length of the listen slots that in our unoptimized implementation is currently set to 125 ms.

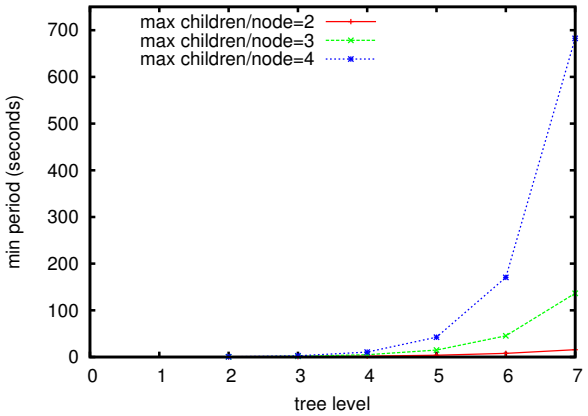

Figure 10. Configurable minimum period depends on the maximum number of children

Figure 10 depicts the minimum period for different tree depths. When the depth, i.e. the number of levels in the tree, increases the minimum period increases exponentially. Note however, that for a maximum number of four children per node, seven levels in the tree correspond to almost 5500 nodes. While it might be possible to re-

duce this minimum period without introducing collisions in particular if the tree is quite sparse we have not considered this option since it might require reconfigurations in case new nodes want to join the network.

For command-response applications, the minimum period doubles compared with the values in Figure 10. If we assume a maximum number of four children per node and a tree with five levels, an operator can still set the sleep period to less than 100 seconds which seems an acceptable trade-off between delay and expected system lifetime.

In order to construct collision-free trees, CoReDac deliberately tolerates unused slots which increases the duration of the collection and command phases in particular when the maximum number of children is large. However, if the maximum is small, there is a danger that sensor nodes are not included in the tree. We have performed initial simulations that have shown that the number of sensors excluded from the network depends very much on the way the network is set up. If it is set up node by node, i.e. one node is started after the previous node is connected to the tree, no node is excluded from the network. This is evident, since the previously connected node per se has empty slots. If all nodes are started at the same time, the sink initiates the setup of the network by sending acknowledgements. Our simulations have shown that in this case having a maximum number of children larger than two is beneficial and a maximum of three leads to reasonable results. The optimal maximum however depends on many factors including the number and density of nodes, exact placement and the number of extra slots reserved for joining nodes. Future work will evaluate different strategies and investigate these trade-offs in more detail.

5 Related Work

In previous work, we have demonstrated that it is possible to implement BACnet on resource-constrained sensor nodes [18]. This paper presents an energy-efficient command-response convergecast protocol that could be used as a MAC layer for this previous work. This scheme would increase the lifetime of such an integrated system.

We have previously underlayed ZigBee with X-MAC and this way increased ZigBee lifetime with a factor of 10 [24]. The experiments in this paper have shown that CoReDac is more power-efficient than X-MAC in convergecast scenarios. X-MAC on the other hand is much more flexible and useful than CoReDac for other scenarios than convergecast.

Dozer [3] is probably the most efficient convergecast protocol. Unlike CoReDac, Dozer explicitly accepts collisions. Dozer's authors claim that a "global TDMA scheme is expensive since it demands the existence of a network-wide time synchronization". Our work shows that this is not the case. Unlike CoReDac and D-MAC, Dozer does not use the notion of a staggered wakeup schedule to decrease data delivery latency.

D-MAC introduced the notion of staggered slots that we also use [13]. Even though D-MAC has introduced some means to reduce the risk for collisions, there is still contention between nodes on the same level and the risk for collision of their data packets. In contrast to this, CoReDac builds a collision-free scheduling scheme based on the notion of staggered slots. Another protocol that avoids but still accepts collisions is DRAND [23]. In contrast to D-MAC, CoReDac also supports an efficient command phase. As CoReDac, wave scheduling is designed to avoid interference between packet transmissions [25]. Wave scheduling is evaluated in the NS-2 simulator only using IEEE 802.11 radios. The same is true for Chiapa et al.'s DCQS protocol [4]. TRAMA is another collision-free MAC protocol that is evaluated by simulation only [22].

Other approaches for convergecast that do not build collision-free trees include Twinkle [11] and the approach proposed by Gandham et al. [10]. The latter tries to reduce latency by minimizing the number of required time slots whereas we deliberately tolerate extra time slots to build CoReDac's collision free trees.

Zhang et al. [28] have focused their convergecast work on data collection only. Their protocol does not include any command phase.

Koala is another approach for low power data retrieval [16]. In this approach, data is not regularly transmitted to the sink but retrieved by bulk data downloads initiated by the sink. Koala and other types of storage-centric sensor networks are useful for sensor network applications that do not require real-time information [14].

6 Conclusions

In this paper, we have presented CoReDac, a convergecast protocol that dynamically builds a collision-free tree based on information in the acknowledgements. In contrast to other convergecast protocols, CoReDac supports a command-response paradigm which makes CoReDac suitable as an underlying layer for e.g. building automation protocols such as BACnet. We have implemented CoReDac in the Contiki operating system. We have demonstrated CoReDac's energy-efficiency by comparing it to X-MAC an energy-efficient MAC protocol for wireless sensor networks. Our experiments on real hardware also demonstrated the low overhead of the command phase.

Acknowledgments

This work was funded by the Swedish Energy Authority and VINNOVA, the Swedish Agency for Innovation Systems.

References

[1] ASHRAE. *BACnet - A Data Communication Protocol for Building Automation and Control Networks*, ansi/ashrae standard 135-2004 edition, 2004.

[2] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *ACM SenSys*, Boulder, USA, Nov. 2006.

[3] N. Burri and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 450–459. ACM Press New York, NY, USA, 2007.

[4] O. Chipara, C. Lu, and J. Stankovic. Dynamic conflict-free query scheduling for wireless sensor networks. In *IEEE International Conference on Network Protocols (ICNP)*, Nov. 2006.

[5] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, Nov. 2004.

[6] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, Nov. 2007.

[7] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32, 2007.

[8] A. El-Hoiydi, J.-D. Decotignie, C. C. Enz, and E. L. Roux. wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In *ACM SenSys (poster proceedings)*, pages 302–303, 2003.

[9] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, Jan. 2007.

[10] S. Gandham, Y. Zhang, and Q. Huang. Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks. *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, July 2006.

[11] B. Hohlt and E. Brewer. Network Power Scheduling for TinyOS Applications. *IEEE Int. Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2006.

[12] W. Kastner, G. Neugschwandtner, S. Soucek, and H. Newmann. Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203, June 2005.

[13] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[14] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. EnviroStore: A Cooperative Storage System for Disconnected Operation in Sensor Networks. In *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, May 2007.

[15] A. Meier, J. Beutel, R. Lim, and L. Thiele. Design of a high-reliability low-power status monitoring protocol. In *Proc. 4th International Conference on Networked Sensing Systems (INSS 2007)*, Braunschweig, Germany, June 2007.

[16] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *The Seventh International Conference on Information Processing in Sensor Networks (IPSN'08)*, 2008.

[17] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.

[18] F. Österlind, E. Pramsten, D. Roberthson, J. Eriksson, N. Finne, and T. Voigt. Integrating building automation systems and wireless sensor networks. In *12th IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, Sept. 2007.

[19] T. Park, Y. Chon, D. Park, and S. Hong. BACnet over ZigBee, A new approach to wireless datalink channel for BACnet. In *5th IEEE International Conference on Industrial Informatics*, June 2007.

[20] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *The Fourth International Conference on Information Processing in Sensor Networks (IPSN/SPOTS)*, Los Angeles, CA, USA, Apr. 2005.

[21] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, Mar. 2002.

[22] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *The First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, California, Nov. 2003.

[23] I. Rhee, A. Warrier, J. Min, and L. Xu. Drand: : distributed randomized tdma scheduling for wireless ad-hoc networks. In *Proceedings of the seventh ACM International Symposium on Mobile ad hoc Networking and Computing*, Florence, Italy, May 2006.

[24] P. Suarez, C.-G. Renmarker, A. Dunkels, and T. Voigt. Increasing ZigBee Network Lifetime with X-MAC. In *Proceedings of the REALWSN'08 Workshop on Real-World Wireless Sensor Networks*, Apr. 2008.

[25] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Wave scheduling and routing in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(1), 2007.

[26] T. Voigt. Self-organizing, collision-free, multi-channel convergecast. In *European Conference on Wireless Sensor Networks (Poster Proceedings)*, Bologna, Italy, Jan. 2008.

[27] T. Voigt, F. Österlind, and A. Dunkels. Improving sensor network robustness with multi-channel convergecast. In *2nd ERCIM Workshop on e-Mobility*, Tampere, Finland, May 2008.

[28] H. Zhang, A. Arora, Y. Choi, and M. Gouda. Reliable bursty convergecast in wireless sensor networks. *Computer Communications*, 30(13):2560–2576, 2007.